Convolutional Neural Network Architectures: from LeNet to ResNet

Lana Lazebnik

Figure source: A. Karpathy
What happened to my field?

**Classification:** ImageNet Challenge top-5 error

Figure source: Kaiming He
What happened to my field?

Object Detection: PASCAL VOC mean Average Precision (mAP)

Figure source: Ross Girshick
Let’s back up even more…

The Perceptron

Input

Weights

\[ x_1, x_2, x_3, \ldots, x_D \]

\[ w_1, w_2, w_3, \ldots, w_D \]

Output: \( \text{sgn}(w \cdot x + b) \)

Two-layer neural network

- Can learn nonlinear functions provided each perceptron has a differentiable nonlinearity

\[ g(t) = \frac{1}{1 + e^{-t}} \]
Multi-layer neural network
Training of multi-layer networks

- Find network weights to minimize the *training error* between true and estimated labels of training examples, e.g.:

\[
E(w) = \sum_{i=1}^{N} (y_i - f_w(x_i))^2
\]

- Update weights by *gradient descent*: \( w \leftarrow w - \alpha \frac{\partial E}{\partial w} \)
Training of multi-layer networks

- Find network weights to minimize the *training error* between true and estimated labels of training examples, e.g.:

\[ E(w) = \sum_{i=1}^{N} (y_i - f_w(x_i))^2 \]

- Update weights by **gradient descent**: \( w \leftarrow w - \alpha \frac{\partial E}{\partial w} \)

- **Back-propagation**: gradients are computed in the direction from output to input layers and combined using chain rule

- **Stochastic gradient descent**: compute the weight update w.r.t. one training example (or a small batch of examples) at a time, cycle through training examples in random order in multiple epochs
“Shallow” vs. “deep” architectures

Traditional recognition: “Shallow” architecture

Image/Video Pixels → Hand-designed feature extraction → Trainable classifier → Object Class

Deep learning: “Deep” architecture

Image/Video Pixels → Layer 1 → ... → Layer N → Simple classifier → Object Class
From fully connected to convolutional networks

image

Fully connected layer
From fully connected to convolutional networks
From fully connected to convolutional networks
From fully connected to convolutional networks
Key operations in a CNN

- Input Image
- Convolution (Learned)
- Non-linearity
- Spatial pooling
- Feature maps

Source: R. Fergus, Y. LeCun
Key operations

Input Image

Convolution (Learned)

Non-linearity

Spatial pooling

Feature maps

Rectified Linear Unit (ReLU)

Source: R. Fergus, Y. LeCun
Key operations

Feature maps

Spatial pooling

Non-linearity

Convolution (Learned)

Input Image

Max

Source: R. Fergus, Y. LeCun
Layer 1 Filters
Layer 1: Top-9 Patches
Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map
Layer 2: Top-9 Patches
Layer 4: Top-9 Patches
Layer 4: Top-9 Patches
Layer 5: Top-9 Patches
Layer 5: Top-9 Patches
Evolution of Features During Training
Evolution of Features During Training
Multi-Layer Network Demo

http://playground.tensorflow.org/
LeNet-5

- Average pooling
- Sigmoid or tanh nonlinearity
- Fully connected layers at the end
- Trained on MNIST digit dataset with 60K training examples

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner,
Fast forward to the arrival of big visual data...

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/
AlexNet: ILSVRC 2012 winner

- Similar framework to LeNet but:
  - Max pooling, ReLU nonlinearity
  - More data and bigger model (7 hidden layers, 650K units, 60M params)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Dropout regularization

Clarifai: ILSVRC 2013 winner

- Refinement of AlexNet

![Diagram of a 8-layer convolutional neural network](image)

Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form \(6 \cdot 6 \cdot 256 = 9216\) dimensions. The final layer is a \(C\)-way softmax function, \(C\) being the number of classes. All filters and feature maps are square in shape.

M. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*, ECCV 2014 (Best Paper Award winner)
VGGNet: ILSVRC 2014 2\textsuperscript{nd} place

- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3x3 convolutions (with ReLU in between)
- One 7x7 conv layer with C feature maps needs $49C^2$ weights, three 3x3 conv layers need only $27C^2$ weights
- Experimented with 1x1 convolutions

<table>
<thead>
<tr>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 layers</td>
<td>11 layers</td>
<td>13 layers</td>
<td>16 layers</td>
<td>16 layers</td>
<td>19 layers</td>
</tr>
</tbody>
</table>

Input (224 × 224 RGB image)

<table>
<thead>
<tr>
<th>A-LRN</th>
<th>A-LRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
</tr>
<tr>
<td>LRN</td>
<td>conv3-64</td>
</tr>
</tbody>
</table>

Maxpool

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
</tbody>
</table>

Maxpool

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
</tbody>
</table>

Maxpool

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
</tbody>
</table>

Maxpool

<table>
<thead>
<tr>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
</tr>
</tbody>
</table>

Maxpool

<table>
<thead>
<tr>
<th>Soft-max</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>FC-4096</td>
</tr>
<tr>
<td>FC-1000</td>
</tr>
<tr>
<td>1x1</td>
</tr>
</tbody>
</table>

Table 2: Number of parameters (in millions).

<table>
<thead>
<tr>
<th>Network</th>
<th>A,A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters</td>
<td>133</td>
<td>133</td>
<td>134</td>
<td>138</td>
<td>144</td>
</tr>
</tbody>
</table>

K. Simonyan and A. Zisserman,
Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
GoogLeNet

- The Inception Module
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps

C. Szegedy et al., *Going deeper with convolutions*, CVPR 2015
ResNet: ILSVRC 2015 winner

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)
VGG, 19 layers (ILSVRC 2014)
ResNet, 152 layers (ILSVRC 2015)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,
Deep Residual Learning for Image Recognition, CVPR 2016
ResNet

- The residual module
  - Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
  - Make it easy for network layers to represent the identity mapping
  - For some reason, need to skip at least two layers

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep Residual Learning for Image Recognition*, CVPR 2016 (Best Paper)
## Summary: ILSVRC 2012-2015

<table>
<thead>
<tr>
<th>Team</th>
<th>Year</th>
<th>Place</th>
<th>Error (top-5)</th>
<th>External data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperVision – Toronto (AlexNet, 7 layers)</td>
<td>2012</td>
<td>-</td>
<td>16.4%</td>
<td>no</td>
</tr>
<tr>
<td>SuperVision</td>
<td>2012</td>
<td>1st</td>
<td>15.3%</td>
<td>ImageNet 22k</td>
</tr>
<tr>
<td>Clarifai – NYU (7 layers)</td>
<td>2013</td>
<td>-</td>
<td>11.7%</td>
<td>no</td>
</tr>
<tr>
<td>Clarifai</td>
<td>2013</td>
<td>1st</td>
<td>11.2%</td>
<td>ImageNet 22k</td>
</tr>
<tr>
<td>VGG – Oxford (16 layers)</td>
<td>2014</td>
<td>2nd</td>
<td>7.32%</td>
<td>no</td>
</tr>
<tr>
<td>GoogLeNet (19 layers)</td>
<td>2014</td>
<td>1st</td>
<td>6.67%</td>
<td>no</td>
</tr>
<tr>
<td>ResNet (152 layers)</td>
<td>2015</td>
<td>1st</td>
<td>3.57%</td>
<td></td>
</tr>
<tr>
<td>Human expert*</td>
<td></td>
<td></td>
<td>5.1%</td>
<td></td>
</tr>
</tbody>
</table>

[http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/]
Accuracy vs. efficiency

https://culurciello.github.io/tech/2016/06/04/nets.html
Design principles

- Reduce filter sizes (except possibly at the lowest layer), factorize filters aggressively
- Use 1x1 convolutions to reduce and expand the number of feature maps judiciously
- Use skip connections and/or create multiple paths through the network
What’s missing from the picture?

• Training tricks and details: initialization, regularization, normalization
• Training data augmentation
• Averaging classifier outputs over multiple crops/flips
• Ensembles of networks

• What about ILSVRC 2016?
  • No more ImageNet classification
  • No breakthroughs comparable to ResNet
Reading list

• [https://culurciello.github.io/tech/2016/06/04/nets.html](https://culurciello.github.io/tech/2016/06/04/nets.html)
• M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
• K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
• M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014
• C. Szegedy et al., Going deeper with convolutions, CVPR 2015
• C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016
• K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016